Interactive Applications

- Interactive applications have response-time requirements
  - Tasks must complete by response time in order to provide good user experience
  - Examples: Video player, user interfaces, games
- Finishing faster than the requirement does not provide any benefit
  - Dynamic voltage and frequency scaling (DVFS) or heterogeneous cores can be used to save energy while meeting response-time requirement
- Opportunity: Applications can show large variations in execution time for each task/input

Feature Generation

- Control flow features (loops, conditionals, and function calls) capture first-order effects on execution time
- Use program slicing to create code fragment for quickly calculating feature values

Prediction Model

- Convex optimization-based linear predictor
  - Minimize: $\|p \cdot (Xb - y)\|^2 + \alpha \|neg(Xb - y)\|^2 + \beta \|b\|_1$
  - Place greater weight on underprediction versus overprediction
    - Underprediction leads to missed deadline
    - Minimize number of features needed

Results

- Evaluation setup
  - ODROID-XU3 development board
  - Compare against existing Linux governors and PID controller
  - Benchmarks include 3 games, a web browser, speech recognition, a video decoder, and two cryptography kernels.
  - Prediction-based controller shows best energy usage and almost no deadline misses

Prediction-Guided DVFS

- Offline: Build prediction model
  - Automatically instrument program for features related to execution time (loop counts, branches taken, etc.)
  - Profile for feature values and execution times
  - Build prediction model from profiling data
  - Create minimal program slice to calculate features and predict execution time
- Run-time: Use prediction to inform DVFS
  - Run program slice to predict execution time
  - Set DVFS based on predicted time and deadline target

Evaluation:

- Overhead of prediction is minimal